Projeto de Bases de Dados
Parte 4

Grupo: 50			
Número	**Nome**	**Contribuição**	**Esforço (em Horas)**
93695	Catarina Sofia dos Santos Sousa	33%	8h
93743	Nelson Alexandre Geada Trindade	33%	8h
93754	Rodrigo Rodrigues Major	34%	8h

Sala: 1-29

Turno: 4ºFeira – 15h00

Docente: Carlota De Oliveira Lopes Dias

Restrições de Integridade

```
--RI-100: um médico não pode dar mais de 100 consultas por se-
mana na mesma instituição
drop trigger if exists verifica_medico_trigger on consulta;

create or replace function verifica_medico() returns trigger as $$
declare consultas decimal(20,2);

begin
  select count(*) into consultas
  from consulta c
  where c.num_cedula = new.num_cedula
  and c.nome_instituicao = new.nome_instituicao
  and EXTRACT(YEAR from c.data) = EXTRACT(YEAR from new.data)
  and EXTRACT(WEEK from c.data) = EXTRACT(WEEK from new.data);

  if consultas >= 100 then
    raise exception 'O médico % não pode dar mais de 100 consultas por se-
mana na mesma instituição.', new.num_cedula;
    end if;
  return new;

 END;
 $$ Language plpgsql;

create trigger verifica_medico_trigger before insert on consulta
for each row execute procedure verifica_medico();

--RI-análise: numa análise, a consulta associada pode estar omissa; não es-
tando, a especialidade
--da consulta tem de ser igual à do médico.
drop trigger if exists verifica_especialidade_trigger on analise;

create or replace function verifica_especialidade() returns trigger as $$
declare especialidade varchar(25);
begin
  select m.especialidade into especialidade
  from consulta c natural join medico m
  where c.num_cedula = new.num_cedula
  and c.num_doente = new.num_doente
  and c.data = new.data;

  if especialidade is not null and especialidade != new.especialidade then
    raise exception 'O médico % não tem a especialidade necessária para ana-
lisar.', new.num_cedula;
    end if;
  return new;
```

```
 END;
 $$ Language plpgsql;

create trigger verifica_especialidade_trigger before insert on analise
for each row execute procedure verifica_especialidade();
```

Índices

Query 1

Não é necessário criar nenhum índice porque é criado implicitamente para as chaves primárias.

Assim, apenas é preciso alterar a ordem dos campos das chaves primárias na declaração da tabela consulta para que o num_doente seja o primeiro atributo. Concluindo, ao alterar a ordem dos campos das chaves primárias, o índice criado implicitamente é o único necessário para acelerar a execução desta query.

Query 2

Como esta query implica uma igualdade no atributo especialidade da tabela medico, então é útil criar um índice que seja apropriado para igualdades. Para isto, criamos um índice do tipo Hash porque funciona por "contentores" e guarda vários registos por cada contentor. Assim, através da função de dispersão o acesso fica otimizado, acelerando a realização desta query.

CREATE INDEX index_especialidade ON medico USING HASH(especialidade)

Query 3

Blocos do disco são de 2KBytes e cada registo ocupa 1kByte, ou seja, cada bloco leva 2 registos. Seletividade de (1/6) = 0.16666667 ou seja, a probabilidade de um bloco não ter respostas é de (1-0.16666667)^2 (por serem dois registos por bloco), que é aproximadamente 69%. Logo, teremos de ler 31% dos blocos. Quanto menor for a percentagem de blocos a ler, maior é o benefício dos índices a reduzir leituras do disco. Assim, como teremos de ler 31% dos blocos, é útil utilizarmos um índice. Uma vez que a query implica uma igualdade no atributo especialidade da tabela medico, então é útil utilizar um índice do tipo Hash para acelerar a realização desta query, porque cada "contentor" aramazena um conjunto de entradas e através da função de dispersão, o acesso fica otimizado.

CREATE INDEX index_especialidade ON medico USING HASH(especialidade)

Query 4

São criados índices implicitamente para as primary keys e assim, pode ser utilizado neste caso para acelerar a execução desta query, uma vez que o atributo num_cedula é o primeiro atributo da chave primária na declaração da tabela consulta. Se o atributo num_cedula não fosse o primeiro da chave primária, então tinhamos de alterar a ordem dos campos das chaves primárias.

Para além deste índice, deve-se criar um índice do tipo BTree para o atributo "data" da tabela consulta para otimizar a comparação entre as duas datas dadas. Este tipo de índice é o apropriado para acelerar esta query porque as folhas do índice estão sempre ordenadas, o que facilita a comparação entre as datas:

CREATE INDEX idx_data on consulta USING B-TREE(data)

Modelo Multidimensional

```
DROP TABLE d_tempo CASCADE;
DROP TABLE d_instituicao CASCADE;
DROP TABLE f_analise CASCADE;
DROP TABLE f_presc_venda CASCADE;

CREATE TABLE d_tempo (
    id_tempo serial PRIMARY KEY,
    dia int,
    dia_da_semana int,
    semana int,
    mes int,
    trimestre int,
    ano int
);

CREATE TABLE d_instituicao (
    id_inst serial PRIMARY KEY,
    nome varchar(255),
    tipo varchar(11),
    num_regiao int,
    num_concelho int,

    FOREIGN KEY (nome) REFERENCES instituicao(nome),
    FOREIGN KEY (num_regiao) REFERENCES regiao(num_regiao),
    FOREIGN KEY (num_concelho) REFERENCES concelho(num_concelho)
);

CREATE TABLE f_presc_venda (
    id_presc_venda int PRIMARY KEY,
    id_medico int,
    num_doente int,
    id_data_registo int,
    id_inst int,
    substancia varchar(255),
    quant int,

    FOREIGN KEY (id_presc_venda) REFERENCES prescricao_venda(num_venda),
    FOREIGN KEY (id_medico) REFERENCES medico(num_cedula),
    FOREIGN KEY (id_data_registo) REFERENCES d_tempo(id_tempo),
    FOREIGN KEY (id_inst) REFERENCES d_instituicao(id_inst)
```

```
);

CREATE TABLE f_analise (
    id_analise int PRIMARY KEY,
    id_medico int,
    num_doente int,
    id_data_registo int,
    id_inst int,
    nome varchar(255),
    quant int,

    FOREIGN KEY (id_analise) REFERENCES analise(num_analise),
    FOREIGN KEY (id_data_registo) REFERENCES d_tempo(id_tempo),
    FOREIGN KEY (id_inst) REFERENCES d_instituicao(id_inst)
);
```

ETL

```
-- d_tempo
With temp as (
  SELECT DISTINCT data from prescricao_venda pv
  UNION
  SELECT DISTINCT data from analise a
)
INSERT INTO d_tempo(dia, dia_da_semana, semana, mes, trimestre, ano)
SELECT extract(day from data),
    extract(dow from data),
    extract(week from data),
    extract(month from data),
    extract(quarter from data),
    extract(year from data)
FROM temp sub;

-- d_instituicao
INSERT INTO d_instituicao(nome, tipo, num_regiao, num_concelho)
SELECT inst.nome,
    inst.tipo,
    inst.num_regiao,
    inst.num_concelho
FROM instituicao inst;

-- f_presc_venda
With temp as (
  SELECT num_cedula, num_doente, id_tempo, substancia, num_venda
  FROM prescricao_venda pv
  INNER JOIN d_tempo dt
```

```
  ON (
    dt.ano = extract(year from pv.data) and
    dt.mes = extract(month from pv.data) and
    dt.dia = extract(day from pv.data)
    )
)
INSERT INTO f_presc_venda(id_presc_venda, id_medico, num_doente, id_data_re-
gisto, id_inst, substancia, quant)
SELECT t.num_venda, num_cedula, num_doente, id_tempo, id_inst, substan-
cia, quant
FROM temp t
INNER JOIN (
  SELECT num_venda, id_inst, quant
  FROM venda_farmacia vf
  INNER JOIN d_instituicao di
  ON vf.inst = di.nome
  ) t2
ON t.num_venda = t2.num_venda;

-- f_analise
with temp as(
  SELECT num_analise, num_cedula, num_doente, id_tempo, inst, nome, quant
  FROM analise a
  INNER JOIN d_tempo dt
  ON (
    dt.ano = extract(year from a.data) and
    dt.mes = extract(month from a.data) and
    dt.dia = extract(day from a.data)
  )
)
INSERT INTO f_analise(id_analise, id_medico, num_doente, id_data_re-
gisto, id_inst, nome, quant)
SELECT num_analise, num_cedula, num_doente, id_tempo, id_inst, t.nome, quant
FROM temp t
INNER JOIN d_instituicao di
ON di.nome = t.inst;
```

Queries OLAP

```sql
--1

SELECT especialidade, mes, ano, count(*)
FROM f_analise a
    INNER JOIN medico ON a.id_medico = medico.num_cedula
    INNER JOIN d_tempo ON a.id_data_registo = d_tempo.id_tempo
WHERE d_tempo.ano >= 2017
    AND d_tempo.ano <= 2020
    AND a.nome = 'glicemia'
GROUP BY CUBE(especialidade, mes, ano)
ORDER BY especialidade, ano, mes ASC;~
```

```sql
--2

WITH TEMP AS(
    SELECT substancia, num_concelho, dia_da_se-
mana, mes, SUM(quant) AS sum_quant, COUNT(*) AS c
    FROM f_presc_venda v
        NATURAL JOIN d_instituicao
        INNER JOIN d_tempo on v.id_data_registo = d_tempo.id_tempo
    WHERE num_regiao = 2
        AND trimestre = 1
        AND ano = 2020
    GROUP BY ROLLUP(substancia, num_concelho, dia_da_semana, mes)
    ORDER BY substancia, mes, dia_da_semana asc
)

SELECT substancia, num_concelho, dia_da_semana, mes, sum_quant,
CASE
WHEN dia_da_semana IS NULL AND mes IS NULL THEN
    CAST(CAST(c AS FLOAT)/90 AS FLOAT(3))

WHEN dia_da_semana IS NOT NULL AND mes IS NULL THEN
    CAST(CAST(c AS FLOAT)/13 AS FLOAT(3))

WHEN dia_da_semana IS NOT null AND mes IS NOT null THEN
    CAST(CAST(c AS FLOAT)/4 AS FLOAT(3))
END AS media
FROM TEMP sub
```